```
inputs = [0.1, 0.2, 0.3]
targets = [0.5, 0.6]
brain.train(inputs, targets)
output = brain.feedforward(inputs)
print("Neural Network Output:", output)
```

2.□□□□□□□□□□□□□□□□

```python
import numpy as np

# □□□□□□□□□
class RoboticArm:
def __init__(self):
self.joints = [0, 0, 0, 0, 0] # 5 □□□□□□

def move_joint(self, joint_index, angle):
if 0 <= joint_index < len(self.joints):
self.joints[joint_index] = angle
print(f"Joint {joint_index} moved to {angle} degrees")
else:
print("Invalid joint index")

def perform_grasp(self):
print("Performing grasp action with current joint angles:", self.joints)

# □□□□□□□□
arm = RoboticArm()
arm.move_joint(0, 30)
arm.move_joint(1, 60)
arm.perform_grasp()
```

3.□□□□□□□□□□□□□

```python
import random

# □□□□□□
class TactileSensor:
def __init__(self):
self.sensitivity = 0.1 # □□□□□□

def read(self):
# □□□□□□□□□□
return random.uniform(0, 1) * self.sensitivity

# □□□□□□□□□□
sensor = TactileSensor()
```

```
for _ in range(5):
print("Tactile Sensor Reading:", sensor.read())
```

4.□□□□□□□□□□□□□□□□□□

```python
import cv2
import speech_recognition as sr

# □□□□
class VisionSystem:
def __init__(self):
self.camera = cv2.VideoCapture(0)

def capture_image(self):
ret, frame = self.camera.read()
if ret:
cv2.imshow("Camera Feed", frame)
cv2.waitKey(1)
return frame
return None

# □□□□
class AuditorySystem:
def __init__(self):
self.recognizer = sr.Recognizer()

def listen(self):
with sr.Microphone() as source:
print("Listening...")
audio = self.recognizer.listen(source)
try:
text = self.recognizer.recognize_google(audio)
print("Heard:", text)
return text
except sr.UnknownValueError:
print("Could not understand audio")
except sr.RequestError as e:
print("Could not request results; {0}".format(e))

# □□□□□□□□□□
vision = VisionSystem()
auditory = AuditorySystem()

# □□□□
vision.capture_image()

# □□□□
auditory.listen()
```

5.□□□□□□□□□□□□□□

```python
class Joint:
def __init__(self, name):
self.name = name
self.angle = 0

def rotate(self, angle):
self.angle = angle
print(f"{self.name} rotated to {angle} degrees")

class Robot:
def __init__(self):
self.neck = Joint("Neck")
self.arm = Joint("Arm")

def move(self, joint_name, angle):
if joint_name == "neck":
self.neck.rotate(angle)
elif joint_name == "arm":
self.arm.rotate(angle)

# □□□□□□□
robot = Robot()
robot.move("neck", 45)
robot.move("arm", 90)
```

6.□□□□□□□□□□□□□□□□□□□□□□□□□□□

```python
from transformers import pipeline

# □□□□□□□□
class LanguageProcessor:
def __init__(self):
self.nlp = pipeline("text-generation")

def generate_response(self, input_text):
response = self.nlp(input_text, max_length=50)
return response[0]["generated_text"]

# □□□□□□□
processor = LanguageProcessor()
response = processor.generate_response("Tell me about artificial intelligence.")
print("Response:", response)
```

```cpp
#include <PID_v1.h>

double Setpoint, Input, Output;
double Kp=2, Ki=5, Kd=1;
PID myPID(&Input, &Output, &Setpoint, Kp, Ki, Kd, DIRECT);

void setup() {
Setpoint = 100; // □□□□
myPID.SetMode(AUTOMATIC); // □□ PID □□□□□□□□
}

void loop() {
Input = readSensor(); // □□□□□□□
myPID.Compute(); // □□ PID □□□□□□
setMotor(Output); // □□□□□□
}
```

8.□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□ Hugging Face Transformers □□□□□□□

```python
from transformers import pipeline

generator = pipeline('text-generation', model='gpt-2')
result = generator("Hello, how are you?", max_length=50)
print(result)
```

9.□□□□□□□□□□□□
□□□□□□□□□□□□ MySQL□□□□□□□□□□□ MongoDB□□□□□□□□□□□□□□□□□□□□□

□□□□□□□□□ MySQL □□□□□□

```python
import mysql.connector

db = mysql.connector.connect(
```

[illegible]
```python
# [illegible][1](it.sohu.com/a/790691508)[illegible]
class BioSkinSensor:
def __init__(self):
self.healing_agent = CollagenGelLayer() # [illegible]
self.nano_sensors = GraphenePressureArray() # [illegible]

def tactile_feedback(self):
# [illegible][9](caifuhao.eastmoney.com/)[illegible]
signal = self.nano_sensors.read_pressure()
return self._neural_processing(signal)

def _neural_processing(self, data):
# [illegible][6](icspec.com/news/article)[illegible]
return spiking_neural_network(data)
```

[illegible]
```python
# [illegible][2](sensorexpert.com.cn/art)[illegible][5](m.sohu.com/a/830923166_)PEEK [illegible]
class DexterousHandController:
def __init__(self):
self.joints = [PEEKJoint() for _ in range(24)] # 24 [illegible]
self.torque_sensors = SixAxisForceTorqueSensor()

def adaptive_grasping(self, object_properties):
# [illegible][4](163.com/dy/article/J9DJ)[illegible]
force_profile = self._calculate_force(object_properties)
for joint in self.joints:
joint.apply_force(force_profile)

def _calculate_force(self, obj):
# [illegible]NVIDIA PhysX [illegible]
return pybullet.calculateInverseDynamics(...)
```

[illegible]
```python
# [illegible][7](news.qq.com/rain/a/2024)[illegible] GPT [illegible][8](news.qq.com/rain/a/2024)[illegible] ARMOR [illegible]
class CognitiveEngine:
def __init__(self):
self.llm = Llama3_220B(finetuned_on="robotic_knowledge")
self.knowledge_graph = Neo4jDatabase() # [illegible]

def reasoning_pipeline(self, sensory_inputs):
# [illegible][9](caifuhao.eastmoney.com/)[illegible]
context = self._context_integration(sensory_inputs)
```

```python
self.neuromorphic_chip = NeuromorphicEngine(config="bio_spiking_nn")

def process(self, data):
# □□□□□□□□
encrypted_data = self.quantum_chip.encrypt(data)
# □□□□□□□□□□□□□□□
result = self.neuromorphic_chip.spike_network(encrypted_data)
return result

# □□□□□□□□□□□□□□□□□□□□□
def _generate_quantum_key(self):
return hashlib.sha256(os.urandom(1024)).hexdigest()
```

---

2. □□□□□□□□□□□□□-□□□□ + □□□□□
```python
class BionicHands:
def __init__(self):
# □□□□□□□□□□□□□□□
self.muscle_driver = HydraulicActuator(pressure=2000) # □□: kPa
# □□□□□□□□□□□□□□
self.tactile_sensors = TactileArray(resolution="1000dpi")

def grasp_object(self, object_shape):
# □□□□□□□□□□□□□□□□□□□□□□
pressure_map = self.tactile_sensors.scan(object_shape)
# □□□□□□□□□□□□□□□
self.muscle_driver.adjust_tension(pressure_map)
return {"status": "grasped", "force": pressure_map.mean()}
```

---

3. □□□□□□□□□□□□□□ + □□□□□□□□
```python
class DeepBrain(nn.Module):
def __init__(self):
super().__init__()
# □□□□□□□□□□□-□□□□-□□□
self.reptilian_layer = nn.LSTM(100, 200) # □□□□
self.limbic_layer = nn.Transformer(200, 100) # □□□□
self.neocortex = nn.ModuleList([nn.Transformer(100, 50) for _ in range(6)]) # □□
□□

def forward(self, x):
# □□□□□□□□□□
x, _ = self.reptilian_layer(x) # □□□
x = self.limbic_layer(x) # □□□
for layer in self.neocortex: # □□□□□□□
x = layer(x) + x # □□□□
return x
```

---

4. □□□□□□□□□□ + □□□□□
```python
class SensorySkin:
def __init__(self):
# □□□□□□□□□□□□□□
self.dermis = SelfHealingPolymer(thickness=2.0) # □□: mm
# □□□□□□□□□□/□□/□□□
self.sensor_net = SensorNetwork(topology="hexagonal")

def detect_threat(self):
# □□□□□□□□□□□□□□
temp = self.sensor_net.read_temperature()
if temp > 60: # □□□
self.dermis.contract() # □□□□□□□
return "DANGER: High temperature"
```

---

5. □□□□□□□□□□□□ + □□□□□
```python
class ChestModule:
def __init__(self):
# □□□□□□□□□□□□
self.modules = {"power": None, "compute": None}
# □□□□□□□□□□□□□
self.auth = HardwareAuth(encryption="ECC-384")

def install_module(self, module):
if self.auth.verify(module.signature):
self.modules[module.type] = module
module.connect_vascular() # □□□□□
```

---

6. □□/□□□□□□□□□ + □□□□□
```python
class VisionSystem:
def __init__(self):
# □□□□□□□□□CMOS□
self.retina = CurvedSensor(fov=160) # □□□□□
# □□□□□□□□□□□□
self.iris = DynamicIris(response_time=0.01) # □□: □

def track_object(self, speed):
# □□□□□□□□□□□
self.iris.adjust(speed)
return self.retina.capture_frame()

class AuditorySystem:
```

```
def __init__(self):
# □□□□□□□□□□□□□□□
self.cochlea = FrequencyBank(resolution="0.1Hz")

def isolate_voice(self, noise_db):
# □□□□□□□□□□□□□□
if noise_db > 80:
self.cochlea.activate_mask("emergency")
```

---

7. □□□□□□□□□□□□□□□□
```python
class BioJoints:
def __init__(self):
# □□□□□□□□□□□□□□
self.ligament = LigamentDamping(factor=0.7)
# □□□□□□□□□IMU□□□
self.balance_ctrl = PIDController(kp=2.5, ki=0.3, kd=1.2)

def walk(self, terrain):
# □□□□□□□□□□□□
if terrain == "stairs":
self._activate_knee_lock() # □□□□□□□
```

---

8. □□□□□□□□□□□□□ + □□□□□
```python
class LanguageSystem:
def __init__(self):
# □□□□□□□□□□□□□□□□GPT□
self.consciousness_engine = ConsciousnessStream()
# □□□□□□□□□□□□□□
self.vocal_cords = BioVocal(emotion_map="pleasure_anger_sadness")

def respond(self, input_text):
# □□□□□□□□
thought = self.consciousness_engine.generate_thought(input_text)
speech = self.vocal_cords.convert_to_speech(thought)
return speech
```

---

9. □□□□□□□□□□□□ + □□□□□
```python
class BrainDatabase:
def __init__(self):
# □□□□□□□□□□□□□□
self.memory = DynamicKnowledgeGraph()
# □□□□□□□□□□□□□□□□
```

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, Embedding

# □□□□□LSTM□□□□
class MindModel:
def __init__(self, input_shape, output_shape):
self.model = Sequential()
self.model.add(LSTM(128, input_shape=input_shape, return_sequences=True))
self.model.add(LSTM(64))
self.model.add(Dense(64, activation='relu'))
self.model.add(Dense(output_shape, activation='softmax'))
self.model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])

def train(self, X_train, y_train, epochs=10):
self.model.fit(X_train, y_train, epochs=epochs)

def predict(self, X_test):
return self.model.predict(X_test)

# □□□□□□□□□
mind_model = MindModel(input_shape=(100, 64), output_shape=10)
mind_model.train(X_train, y_train, epochs=20)
```

2. □□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□OpenAI□Gym□□□□□□□□□□

```python
import gym
import numpy as np

# □□□□□□□□□□
class HandController:
def __init__(self, env_name='HandReach-v0'):
self.env = gym.make(env_name)
self.state_size = self.env.observation_space.shape[0]
self.action_size = self.env.action_space.shape[0]

def train(self, episodes=1000):
for episode in range(episodes):
state = self.env.reset()
done = False
while not done:
action = self.env.action_space.sample() # □□□□
next_state, reward, done, _ = self.env.step(action)
state = next_state

# □□□□□□□□□□
hand_controller = HandController()
hand_controller.train(episodes=1000)
```

```
def perform_task(self):
print(f"{self.module_name} is performing its task.")

# □□□□□□□□
power_module = ChestModule("Power Module")
sensor_module = ChestModule("Sensor Module")
power_module.perform_task()
sensor_module.perform_task()
```

6. □□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ OpenCV □ SpeechRecognition □□□□□□□□□□□□

```python
import cv2
import speech_recognition as sr

# □□□□
class VisionSystem:
def __init__(self):
self.camera = cv2.VideoCapture(0)

def capture_image(self):
ret, frame = self.camera.read()
return frame

# □□□□
class HearingSystem:
def __init__(self):
self.recognizer = sr.Recognizer()

def recognize_speech(self, audio_file):
with sr.AudioFile(audio_file) as source:
audio = self.recognizer.record(source)
return self.recognizer.recognize_google(audio)

# □□□□□□□□□□
vision_system = VisionSystem()
image = vision_system.capture_image()
cv2.imwrite("captured_image.jpg", image)

hearing_system = HearingSystem()
text = hearing_system.recognize_speech("audio_file.wav")
print("Recognized Text:", text)
```

7. □□□□□□□□□□
□□□□□□□□□□□□□□□□□□□Inverse Kinematics□□□□□□□□□□□ PyBullet □□□□□□□□□□□□

```python
import pybullet as p

# □□□□□□
```

```
__tablename__ = 'knowledge'
id = Column(Integer, primary_key=True)
fact = Column(String)
context = Column(String)

# □□□□□
engine = create_engine('sqlite:///knowledge.db')
Base.metadata.create_all(engine)
Session = sessionmaker(bind=engine)
session = Session()

# □□□□□□□
new_knowledge = Knowledge(fact="□□□□□□□", context="□□□□")
session.add(new_knowledge)
session.commit()

# □□□□
knowledge = session.query(Knowledge).filter_by(fact="□□□□□□□").first()
print(knowledge.context)
```

10. □□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ Python □□□□□□□□□□□□

```python
import threading
import time

# □□□□□□□□□
class BackupSystem:
def __init__(self):
self.backup_data = []

def backup(self, data):
self.backup_data.append(data)
print(f"Backup: {data}")

# □□□□□□□□
backup_system = BackupSystem()

def task(data):
backup_system.backup(data)

threads = []
for i in range(5):
t = threading.Thread(target=task, args=(f"Data {i}",))
threads.append(t)
t.start()

for t in threads:
t.join()
```

□□

else:
print("Invalid input")

def grasp_object(self, pressure):
print(f"Grasping with pressure {pressure}")

# □□
hand = RoboticHand()
hand.move_finger(0, 50) # □□□□□□□□□□ 50
hand.grasp_object(75) # □□□□□□□□ 75
```

---

### ③ **□□□□□□**
□□□□□□□□□□□□□□□□□□□□□□□

```python
import numpy as np

class Brain:
def __init__(self):
self.memory = [] # □□□□

def store_memory(self, event):
self.memory.append(event)
print(f"Stored memory: {event}")

def recall_memory(self):
return np.random.choice(self.memory) if self.memory else "No memory stored"

# □□
brain = Brain()
brain.store_memory("Learned to grasp objects")
print(brain.recall_memory())
```

---

④ **□□□□**
□□□□□□□□□□□□□□□□□□□□□□□□

```python
class SensorySkin:
def __init__(self):
self.temperature = 25.0 # □□□□
self.pressure = 0.0 # □□□□

def update_sensors(self, temp, press):
self.temperature = temp
self.pressure = press
print(f"Updated sensors: Temperature={temp}, Pressure={press}")

# □□

```
skin = SensorySkin()
skin.update_sensors(30.5, 10.2)
```

---

⑤ **□□□□□□□□□**
□□□□□□□□□□□□□□□□□

```python
class ChestModule:
def __init__(self):
self.modules = []

def add_module(self, module):
self.modules.append(module)
print(f"Added module: {module}")

def remove_module(self, module):
if module in self.modules:
self.modules.remove(module)
print(f"Removed module: {module}")

# □□
chest = ChestModule()
chest.add_module("Power Supply")
chest.add_module("Cooling System")
```

---

⑥ **□□□□□□□□□**
□□□□□□□□□□□□□□□

```python
import cv2
import speech_recognition as sr

class VisionHearing:
def __init__(self):
self.recognizer = sr.Recognizer()

def capture_image(self):
camera = cv2.VideoCapture(0)
ret, frame = camera.read()
cv2.imwrite("captured_image.jpg", frame)
camera.release()
print("Image captured")

def recognize_speech(self):
with sr.Microphone() as source:
audio = self.recognizer.listen(source)
try:
text = self.recognizer.recognize_google(audio)
```

```
print(f"Recognized speech: {text}")
except sr.UnknownValueError:
print("Could not understand audio")

# □□
vh = VisionHearing()
vh.capture_image()
vh.recognize_speech()
```

---

⑦ **□□□□□□□□**
□□□□□□□Inverse Kinematics□□□□□□□□

```python
import numpy as np

class ArmController:
def __init__(self):
self.joint_angles = [0, 0, 0] # □□□□□□□

def move_arm(self, target_position):
# □□□□□□□□□
self.joint_angles = np.arctan2(target_position[1], target_position[0])
print(f"Arm moved to angles: {self.joint_angles}")

# □□
arm = ArmController()
arm.move_arm([1, 1])
```

---

⑧ **□□□□□□□□□□□□□□□□□□□**
□□□□□□□□□NLP□□□□□□□

```python
from transformers import pipeline

class LanguageSystem:
def __init__(self):
self.nlp = pipeline("text-generation")

def generate_response(self, prompt):
response = self.nlp(prompt, max_length=50)
print(f"Generated response: {response}")

# □□
lang_sys = LanguageSystem()
lang_sys.generate_response("Hello, how are you?")
```

Design and development of super-advanced intelligent humanoid robot design program code, including mechatronics, automatic control servo driver, detailed design of core components, software and hardware system, brain design, body diary, limbs, especially the five fingers of the hand, are flexible to 50 degrees. The neck is flexible, hands and feet are equally important, you can talk to yourself and communicate with human beings, the movements of the five senses and limbs are developed, you can sleep, get up, run and dance freely, weigh 60/70 kilograms, have different models, and have a height of 1.5/1.6 meters. It can be used continuously for 24 hours with long-acting livestock batteries. It is made of composite metal materials, precision machinery, miniaturization, lightweight, durability, standardization, universality, technical redundancy, safety and practicality. Multi-function and multi-purpose, suitable for daily life, work, study, labor, entertainment and sports, etc., bionic simulation, truly integrating man and machine, advanced intelligent robot exceeds all kinds of humanoid robots at home and abroad, and the manufacturing cost is 50-100 thousand yuan, the popular type is 300-500 thousand yuan, and the advanced model is 100-200-5 million yuan, which is suitable for commercial production. ●● Price (1) 50,000-100,000 yuan for low level, 100,000-200,000 yuan for intermediate level, 300,000-500,000 yuan for ordinary high level, 500-1,000,000 yuan for advanced level, 1,000-5,000,000 yuan for export of high-end type, and 2,000-5,000,000 dollars for ultra-advanced intelligent robot design. Because it involves interdisciplinary complex system integration and trade secret protection, it is The following is a detailed analysis of the technical framework and core modules: 1. Design of mechatronics system 1. Drive system architecture-micro harmonic reduction motor (torque density ≥50Nm/kg)- three-stage planetary gear transmission system (transmission efficiency > 92%)- bionic tendon structure (carbon fiber -SMA composite material, Strain rate 0.5-1.2 mm/) 2. Automatic control system 1. Servo drive module-dual DSP architecture (Titms 320F28379D+Xilinx ZNQ ultrascale+MPSOC)-adaptive PID algorithm (response time < 0.8ms)- six-axis force feedback system (resolution 0.01N·m) 3. Bionic motion system 1. Hand mechanism-5-DOF modular finger (bending angle 52 0.5)-piezoelectric tactile sensor array (4096 points/cm)- variable stiffness mechanism (0.5-5N/mm continuous adjustment) 4. Intelligent interactive system 1. Multi-modal interactive engine-hybrid dialogue system (GPT-4 architecture+domain knowledge map)-micro-expression generation system (72 groups of facial actuation units)-multi-channel perception fusion (lidar+millimeter wave +3D structured light) 5. Energy and power system 1. High-density energy module-solid lithium-sulfur battery pack (energy density 600Wh/kg)- wireless charging system (wireless charging system) Efficiency 85%)- Energy recovery device (kinetic energy conversion rate > 30%) VI. Safety redundancy design 1. Triple fault-tolerant architecture-Three-mode redundancy of main control chip (Lockstep architecture)-Emergency braking system (response time < 5ms)- Self-check diagnosis module (500+ health status parameters) VII. Cost control scheme 1. Mass production optimization strategy-modular design (generalization rate > 75%)- mixed manufacturing process (3D printing+precision casting)-hierarchical management of supply chain (autonomy rate of core components is 60%) Note: The specific implementation needs to comply with robot safety standards such as ISO 13482 and ISO/TC 299, and it is recommended to adopt phased development strategy: 1. Prototype verification stage (18 months, Investment of 5-8 million) 2. Engineering prototype stage (12 months, investment of 12-15 million) 3. Mass production preparation stage (6 months, production line investment of 30 million+). It is suggested to give priority to the development of core control algorithms and drive systems and establish patent barriers (20+ invention patents can be applied). If further

technical details are needed, it is suggested to form an interdisciplinary team (experts in the fields of machinery, electronics, AI, materials, etc.) to carry out special research. ●1. Technical feasibility analysis:-At present, the single cost of the world's most advanced biped robot (such as Boston Dynamics Atlas) exceeds 2 million US dollars.-The single price of miniaturized high-precision servo motors (such as maxon EC series) is 3,000-8,000 yuan.-High-energy-density batteries (such as Tesla 4680) still need to break through to reach 24-hour battery life. 2. Suggestions on the architecture of core technology modules. : [Hardware system]-Bionic skeleton: titanium alloy 3D printing skeleton (40% weight reduction)-Driving system: self-made magnetically coded servo motor (0.05 accuracy)-Sensing system: multi-modal sensor fusion array (16 channels IMU+3D LiDAR)[Software architecture] -Real-time operating system: ROS2+Linux RT kernel-Motion control: gait generation algorithm based on reinforcement learning-Human-computer interaction: multi-modal large language model (localized deployment) 3. Implementation example of key technologies (Python pseudocode): ``` python# Multi-joint motion control algorithm class biomimetic controller: def _ _ init _ _ (self): self. muscle _ model = hilltypemodel () self. balance _ ctrl = zmpcontroller () def dynamic_balance(self, sensor_data): com = self.calculate_center_of_mass() zmp = self.balance_ctrl.compute_zmp(com, sensor_data) torque = self.muscle_model.compute_torque(zmp) return self.apply_torque_distribution(torque)# The core logic of dialogue system is classcognitive engine: defprocess _ input (self, text, vision, audio): context = self. multimodal _ fusion (text, vision, audio) intent = self.intent_recognition(context) return self.knowledge_graph.query(intent)``` 4. Suggestions on the commercial realization path:-The first stage (2-3 years): developing the basic platform of education version (the price is 80,000 yuan)-The second stage (5 years): realizing medical care/ Service Professional Edition (priced at RMB350,000)-The third stage (8-10 years): Complete the fully functional humanoid robot. 5. Safety and ethical considerations:-Need to pass the ISO 13482 safety certification-Set the three-level physical safety switch-Follow the AI ethical standards (including the extended version of Asimov's Three Principles). The research and development of this level of robot needs to set up a team with at least the following experts:-Mechanical engineer (5 people+)-Control algorithm expert. -AI researcher (3 people+)-Electronic engineer (2 people+)-Industrial designer (2 people+) suggested giving priority to the development of application versions of specific scenarios (such as rehabilitation robots) and gradually evolving to general use. The complete technical scheme needs to be obtained through authentication channels after signing NDA. Under the current technical conditions, it is suggested to adopt modular design ideas, focusing on breaking through the two core problems of dexterous hand operation and dynamic balance control. Specific technical documents can refer to the latest research results of IEEE Humanoid Robotics. Designing an ultra-advanced intelligent robot involves many complex engineering fields, including mechanical design, electronic engineering, software programming, artificial intelligence, material science and so on. ### 1. Electromechanical integration design is the core of robot design, involving the integration of mechanical structure, electronic control and sensor system. # # # 1.1 Mechanical structure design-* * Body structure * *: Lightweight composite metal materials (such as aluminum alloy and titanium alloy) and carbon fiber composite materials are adopted to ensure strength and lightweight. -* * Limb design * *: The limbs are designed in a modular way, and the joints use high-precision servo motors and reducers to ensure flexibility and load capacity. -* * Hand design * *: Five-finger design adopts bionics principle, each finger has three joints, and micro servo motor and flexible sensor are used to realize highly flexible operation ability (above 50

degrees). -* * Neck design * *: The neck is designed with multiple degrees of freedom, and servo motors and precision bearings are used to ensure flexible rotation and stability. # # # 1.2 Electronic control system-* * Servo driver * *: High-performance servo driver is adopted to support high-precision position control and torque control. -* * Sensor system * *: including force sensor, gyroscope, accelerometer, visual sensor (camera), infrared sensor, etc., used for environmental perception and motion control. -* * Power management * *: Long-lasting lithium battery pack is adopted to support 24-hour continuous use, and intelligent power management system is equipped to optimize energy consumption. ### 2. Automatic control and servo driver-* * Motion control algorithm * *: PID control algorithm or more advanced model predictive control (MPC) algorithm is used to ensure the accuracy and stability of robot motion. -* * Servo driver * *: It adopts digital servo driver and supports CAN bus or EtherCAT communication protocol to realize high-speed and high-precision motion control. ### 3. Detailed design of core components-* * Servo motor * *: Miniaturized design, high power density and fast response. -* * Reducer * *: Use harmonic reducer or planetary reducer to ensure high torque output and low backlash. -* * Sensor module * *: It integrates various sensors to realize multi-modal sensing. ### 4. Software system design # # # 4.1 Operating system-Real-time operating system (RTOS) such as FreeRTOS or ROS (Robot Operating System) is adopted to ensure real-time performance and multi-task processing ability. #### 4.2 Artificial Intelligence and Machine Learning-* * Speech Recognition and Synthesis * *: Use deep learning model (such as Transformer) to realize natural language processing (NLP), which supports soliloquy and human interaction. -* * Computer Vision * *: Convolutional Neural Network (CNN) is used for image recognition and target tracking. -* * Motion planning * *: Use reinforcement learning (RL) algorithm for motion planning and optimization. #### 4.3 Control software-* * Motion control module * *: realize the motion control of the robot, including gait generation and balance control. -* * Task scheduling module * *: manages the robot's task execution and supports multi-task parallel processing. ### 5. Hardware system design-* * Main control unit * *: High-performance embedded processor (such as ARM Cortex-A series or NVIDIA Jetson series) is adopted to support multi-core parallel computing. -* * Communication module * *: supports communication protocols such as Wi-Fi, Bluetooth and 5G, and ensures the seamless connection between the robot and external devices. -* * Storage module * *: High-speed solid state drive (SSD) is adopted to ensure data storage and reading speed. ### 6. Brain design-* * Neural network architecture * *: Deep neural network (DNN) and recurrent neural network (RNN) are adopted to realize advanced cognitive function and decision-making ability. -* * Memory module * *: The distributed storage system is used to support long-term memory and short-term memory. ### 7. Body design-* * Appearance design * *: Bionics design is adopted, the appearance is close to that of human beings, and composite metal and flexible materials are used to ensure beauty and durability. -* * Internal structure * *: Modular design, easy to maintain and upgrade. ### 8. Diary function-* * Log record * *: The robot can automatically record daily activities and interaction information and store it in a local log file. -* * Data analysis * *: Use machine learning algorithm to analyze the log data and optimize the behavior mode of the robot. ### 9. Limb design-* * Hand design * *: Five fingers are highly flexible, each finger has three joints, and micro servo motors and flexible sensors are used to support fine operation. -* * Foot design * *: bionic design is adopted to support complex sports such as running and dancing. ### 10. Design of five senses-* * Vision system * *: Use high-definition camera and depth sensor to support 3D vision and environmental perception. -* * Auditory system * *: Using microphone array and speech